VSP Public Comment

From: Alan Dechert [alan@openvotingconsortium.org]

Sent: Friday, July 01, 2005 10:40 AM

To: Jefferson, David

Cc: Dick Johnson; Jessica D Waselkow; McDannold, Bruce

Subject: Re: Argument against public source software for voting systems

The argument presented here is not compelling. The three "consequences of such variant voting system codebases being around" apply equally -- or even more so -- for closed source voting systems. Public source systems gain scrutinizers, while attack vectors remain the same as closed source systems.

Specifically, this reasoning is incorrect:

> "... because the source is public, there can be thousands of variants already circulating, so the barrier to attempting this fraud is much reduced."

It doesn't matter how many variants circulate. For a given version of a product, only one codebase will be certified. Methods to authenticate the version in use are well known. If correct methods are followed, we can be confident that the certified version is running in production. If authentication methods are lax, then "Anyone with access to the distribution tree for voting system software could substitute ..." a hacked version. While the open source model may make it easier to find a version to substitute, it would not necessarily make it easier to do so without detection.

Also, consider that "open voting systems" means more than public source software. It means that all aspects of election administration be open to public scrutiny. If the principles of open voting are in force, we will have a public record of who did what, when, where, how, and by what authority. So malicious substitutions are less likely than where "security through obscurity" principles apply.

> "We simply cannot permit public source for voting system code at least until we have a rock-solid secure code distribution system."

If we accept this reasoning for public source systems, why should we excuse closed source systems from the same imperative? No case has been made for excusing closed source systems in this regard. If we really take this to heart, then, logically, we'd should go back to placing pebbles in an urn or some other low tech method for voting until we upgrade the election software distribution system.

Furthermore, the barrier to such fraud also depends on the strength of other auditing methods. For example, manually checking a truly random sample of paper ballots against the electronic tally will reveal the introduction of malicious hacks in the tabulation software. Proper audit procedures should also reveal where the malicious code was introduced, and by whom. This has nothing to do with whether or not the source is public.

In total, the essay under consideration serves as a weak argument for continuing with a broken closed source software distribution model. The underlying chain-of-custody issues are not discussed.

Who is supposed to do what, and by what authority? Can we find some election code that says something like, "The Secretary of State shall establish, execute, or supervise procedures to ensure that every computer software application used in elections is identical to the appropriate certified version?" I believe election law allows the Secretary of State to implement some security measures, but it does not pin responsibility on anyone. Who is at fault if malicious code is substituted?

Since the Secretary of State ultimately certifies the election result for the state, responsibility for the correctness of the procedures -- computerized or manual -- accretes there. Furthermore, by virtue of her roles as system certifier and holder of funds for purchasing systems, the Secretary of State has gained a broad responsibility and authority over the voting system. We need to identify and track how authority flows from the Secretary of State to the counties. At what points does the Secretary of State transfer responsibility? How do people downstream receive authorization to perform actions within the voting process?

Occasionally, responsibility is formalized in court cases. But in large measure, defining authority and responsibility has been heuristic and informal. Accountability is an issue. When things go wrong, who is responsible and what are the consequences?

Developing a secure code distribution system is just one aspect of formalizing the chain-of-custody/authority/responsibility

throughout the voting system. Keeping source code secret does not help this process.
Alan Dechert

----- Original Message -----
From: David Jefferson
To: Alan Dechert
Cc: David Jefferson
Sent: Saturday, June 25, 2005 11:37 AM
Subject: Help: Argument against public source software for voting systems

Alan,

I have circulated the following argument to a few people for comment and I wanted to get yours as well. It is an argument against quick adoption of public or open source software, at least for the time being. I am actually looking for strong refutations--don't worry that I am changing mind about open source. But the argument need to be dealt with. Please tell me what you think.

The strength of the public source voting system software idea is that anyone can read the code, learn how it works, criticize it, and scrutinize it for bugs, bias, malicious logic, etc. In a democracy, the public has a right to know every detail about how their votes are collected and processed. Losing candidates have a right to inspect the software to see if they are the victims of bugs. Political parties have the right to look for any possibility of bias in the logic of the code. Everyone has the right to check to see if the code inadvertantly leaks information about how they vote. The disabled have a right to verify that their votes are treated the same as everyone else's. Experts should be able to judge and criticize the quality of the engineering and security. And everyone should be able to judge for himself the vulnerability of the code to attacks. Keeping the source code secret is tantamont to keeping the mechanics of democracy secret; it makes voting systems black boxes. Over time public confidence in elections will inevitably erode if the software is kept secret.

But...

If the code is made public, tens of thousands of people can download copies, and modify their copies, and rebuild the executables to insert malicious variations. Most will be made by bored teenagers, but some may be created by serious conspiracies or enemy agencies. Thousands of these variations may be circulating above ground and underground, and it will be impossible to collect or categorize them all.

Now, consider the consequences of such variant voting system codebases being around.

1) Anyone with access to the distribution tree for voting system software could substitute one of these circulating variants and, depending on what software is affected and where in the tree the substitution is made, a single machine could be affected, or a precinct, or a county, or a state, or the entire country. Some variants might be specifically targeted at Florida, or Ohio, for example. Lots of people might take up the challenge to create these variants. Although it would be illegal, it would be impossible to enforce.

Now, if you insert a variant of closed source software into the distribution tree the same thing can happen; but the difference is that because the source is public, there can be thousands of variants already circulating, so the barrier to attempting this fraud is much reduced. The person who writes the malicious modification in the first place does not need to breach the vendor's security to get access to the source, and never has to meet the person who inserts the code into the distribution chain. In other words, the barriers to fraud a fewer, and different people can do different parts of the job *without being a conspiracy.*

2) This danger is real even if there is VVPT, because we know of powerful malicious presentation attacks and also vote privacy attacks that VVPT does not address.

3) Of course, if there is a cryptographic check on the code as it is loaded into a voting machine, such attacks can be defeated, but we don't have anything like that at all now, and I bet there is nothing mentioned in the TGDC draft about it. (I should check I guess.)

It seems to me that all of this argues that we simply cannot permit public source for voting system code *at least until we have a rock-solid secure code distribution system.*

Comments?

David